IN THE CLAIMS:

1. (currently amended) A superscalar processor for performing out of order processing on an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith, the superscalar processor comprising:

at least one execution unit for executing a plurality of in-flight instructions of the instruction set;

a plurality of registers;

a fetch unit for fetching the instructions from the instruction set;

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set wherein the instruction set comprises one or more instructions whose data sources and destinations are registers;

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of all the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data and operation codes necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed; and

a content addressable memory (CAM) structure having a comparator section mapped to an array section, the CAM structure transmitting a first output signal for generating the dependency data stored in the dependency matrix and a second output signal for generating the physical register address data stored in the IWB.

2. (original) The superscalar processor of claim 1 wherein the CAM structure further comprises a plurality of CAM structures each having a first and second output signal, the first output signals being logically OR-ed together at an ORing unit to generate the dependency data.

3-11. (cancelled)

12. (original) A superscalar processor for performing out of order processing on an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith, which have a first predetermined maximum number of register source fields (RS fields) and at least a maximum number of one register destination field (RD field), the superscalar processor comprising:

at least one execution unit for executing a plurality of in-flight instructions of the instruction set;

a plurality of registers;

a fetch unit for fetching the instructions from the instruction set;

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set;

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed; and

a first predetermined number of content addressable memory (CAM) structures having a comparator section mapped to an array section, each CAM structure being dedicated to a single RS field and transmitting a first and second output signal, the first output signals of each CAM structure being logically OR-ed together at an ORing unit for generating the dependency data stored in the dependency matrix and the second output signals for each CAM structure being combined for generating the physical register address data stored in the IWB.

13. (original) The superscalar processor of claim 12 wherein the RS fields have architectural addresses for instruction input data and the RD fields have architectural addresses for instruction output data, and wherein the CAM structures further comprise:

the comparator sections including a register dependency checker (RDC) for comparing an RS field of a fetched instruction to an RD field of an in-flight instruction;

the first output signals including a hit detect signal indicative of the architectural address of the RD field which is generated when a match, i.e. hit, between the RS field

and the RD field is detected by the RDC, the hit detect signal additionally being transmitted to the array section of the CAM structures;

the array sections including an in-flight physical register mapper (IPM); and

the second output signals including an output of the IPM indicative of the physical register address of the RD field which is generated when the IPM receives the hit detect signal.

14. (original) The superscalar processor of claim 13 further comprising:

the plurality of physical registers including a queue area for storing a second predetermined maximum number of in-flight instructions of the instruction set;

the fetch unit fetching a bundle having a third predetermined fixed number of instructions from the instruction set; and

the RDC further including a plurality of the second predetermined number of entry structures, each entry structure including a plurality of the third predetermined number of comparators, each comparator dedicated to a single instruction in the bundle; wherein each RD field of the in-flight instructions is compared by a single entry structure to all instructions in the bundle substantially simultaneously to generate the hit detect signal.

15. (currently amended) A computer system comprising:

a bus;

an input/output subsystem for interfacing with input/output devices;

a memory subsystem having a memory for storing an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith wherein the instruction set comprises one or more instructions whose data sources and destinations are registers; and

A superscalar processor for performing out of order processing on the instruction set and being in communication with the input/output system and the memory system through the bus, the superscalar processor including,

at least one execution unit for executing a plurality of in-flight instructions of the instruction set,

a plurality of physical registers,

a fetch unit for fetching the instructions from the instruction set,

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set,

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of all the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data and operation codes necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed, and

a content addressable memory (CAM) structure having a comparator section mapped to an array section, the CAM structure transmitting a first output signal for generating the dependency data stored in the dependency matrix and a second output signal for generating the physical register address data stored in the IWB.

16. (original) The computer system of claim 15 wherein the CAM structure further comprises a plurality of CAM structures each having a first and second output signal, the first output signals being logically OR-ed together at an ORing unit to generate the dependency data.

17-25. (cancelled)

26. (currently amended) A method for performing out of order processing on an instruction set having a plurality of instructions with a superscalar processor, the method comprising:

fetching the instructions from the instruction set wherein the instruction set comprises one or more instructions whose data sources and destinations are registers;

renaming architectural registers associated with the instructions to physical registers during the processing of the instruction set;

transmitting a first output signal from a content addressable memory (CAM) structure for generating dependency data of all in-flight instructions of the instruction set;

transmitting a second output signal from the CAM structure for generating physical register address data necessary to execute the in-flight instructions when the dependencies of the in-flight instructions have been removed;

storing the dependency data in a dependency matrix;

storing the physical register address data and operation codes in an instruction wait buffer;

scheduling the in-flight instructions for execution based on the dependency data and the physical register address data; and

executing the in-flight instructions.

27. (original) The method of claim 26 wherein transmitting the first output signal further comprises:

transmitting a plurality of first output signals from a plurality of CAM structures; and

logically ORing the first output signals to generate the dependency data.

28-32. (cancelled)

33. (new) A superscalar processor for performing out of order processing on an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith, the superscalar processor comprising:

at least one execution unit for executing a plurality of in-flight instructions of the instruction set;

a plurality of registers;

a fetch unit for fetching the instructions from the instruction set;

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set;

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data

necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed; and

a content addressable memory (CAM) structure having a comparator section mapped to an array section, the CAM structure transmitting a first output signal for generating the dependency data stored in the dependency matrix and a second output signal for generating the physical register address data stored in the IWB;

wherein the CAM structure further comprises a plurality of CAM structures each having a first and second output signal, the first output signals being logically OR-ed together at an ORing unit to generate the dependency data;

wherein the instructions have a first predetermined maximum number of register source fields (RS fields) and at least a maximum number of one register destination fields (RD fields); and

wherein the CAM structures further comprise the first predetermined maximum number of CAM structures, each CAM structure being dedicated to a single RS field.

34. (new) A superscalar processor for performing out of order processing on an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith, the superscalar processor comprising:

at least one execution unit for executing a plurality of in-flight instructions of the instruction set;

a plurality of registers;

a fetch unit for fetching the instructions from the instruction set;

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set;

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed; and

a content addressable memory (CAM) structure having a comparator section mapped to an array section, the CAM structure transmitting a first output signal for

generating the dependency data stored in the dependency matrix and a second output signal for generating the physical register address data stored in the IWB;

wherein the instructions have a first predetermined maximum number of register source fields (RS fields) having an architectural addresses for instruction input data and at least a maximum number of one register destination fields (RD fields) having architectural addresses for instruction output data; and

wherein the CAM structure further comprises:

the comparator section including a register dependency checker (RDC) for comparing an RS field of a fetched instruction to an RD field of an in-flight instruction; and

the first output signal including a hit detect signal indicative of the architectural address of the RD field which is generated when a match, i.e. hit, between the RS field and the RD field is detected by the RDC, the hit detect signal additionally being transmitted to the array section of the CAM structure.

35. (new) The superscalar processor of claim 34 wherein the CAM structure further comprises:

the array section including an in-flight physical register mapper (IPM); and

the second output signal including an output signal indicative of the physical register address of the RD field which is generated when the IPM receives the hit detect signal.

36. (new) The superscalar processor of claim 34 wherein the CAM structure further comprises a plurality of the first predetermined number of CAM structures, each CAM structure being dedicated to a single RS field.

37. (new) The superscalar processor of claim 36 wherein the hit detect signals of each CAM structure are OR-ed together in an ORing unit to generate the dependency data.

38. (new) The superscalar processor of claim 37 further comprising:

the plurality of physical registers including a queue area for storing a second predetermined maximum number of in-flight instructions of the instruction set; and

the fetch unit fetching a bundle having a third predetermined fixed number of instructions from the instruction set.

39. (new) The superscalar processor of claim 38 further comprising an intra dependency checker having an output signal indicative of the dependencies between the instructions in the bundle, the output signal of the intra dependency checker being combined with the OR-ed output signals of the hit detect signals to generate the dependency data.

40. (new) The superscalar processor of claim 39 wherein the RDC further comprises:

a plurality of the second predetermined number of entry structures, each entry structure including a plurality of the third predetermined number of comparators, each comparator dedicated to a single instruction in the bundle; wherein each RD field of the in-flight instructions is compared by a single entry structure to all instructions in the bundle substantially simultaneously.

41. (new) The superscalar processor of claim 40 wherein the hit detect signal further comprises an array of the second predetermined number by the third predetermined number in size, which includes the output of hits detected for each of the entry structures compared to each of the instructions in the bundle.

42. (new) A computer system comprising:

a bus;

an input/output subsystem for interfacing with input/output devices;

a memory subsystem having a memory for storing an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith; and

A superscalar processor for performing out of order processing on the instruction set and being in communication with the input/output system and the memory system through the bus, the superscalar processor including,

at least one execution unit for executing a plurality of in-flight instructions of the instruction set,

a plurality of physical registers,

a fetch unit for fetching the instructions from the instruction set,

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set,

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed, and

a content addressable memory (CAM) structure having a comparator section mapped to an array section, the CAM structure transmitting a first output signal for generating the dependency data stored in the dependency matrix and a second output signal for generating the physical register address data stored in the IWB;

wherein the CAM structure further comprises a plurality of CAM structures each having a first and second output signal, the first output signals being logically OR-ed together at an ORing unit to generate the dependency data;

wherein the instructions have a first predetermined maximum number of register source fields (RS fields) and at least a maximum number of one register destination fields (RD fields); and

wherein the CAM structures further comprise the first predetermined maximum number of CAM structures, each CAM structure being dedicated to a single RS field.

43. (new) A computer system comprising:

a bus;

an input/output subsystem for interfacing with input/output devices;

a memory subsystem having a memory for storing an instruction set having a plurality of instructions and a plurality of architectural registers associated therewith; and

A superscalar processor for performing out of order processing on the instruction set and being in communication with the input/output system and the memory system through the bus, the superscalar processor including,

at least one execution unit for executing a plurality of in-flight instructions of the instruction set,

a plurality of physical registers,

a fetch unit for fetching the instructions from the instruction set,

an instruction renaming unit (IRU) for renaming architectural registers to physical registers during the processing of the instruction set,

an instruction scheduling unit (ISU) for scheduling the in-flight instructions for execution, the ISU including a dependency matrix for storing dependency data of the in-flight instructions and an instruction wait buffer (IWB) for storing physical address data necessary to execute the in-flight instructions when the dependency matrix indicates that the dependencies of the in-flight instructions have been removed, and

a content addressable memory (CAM) structure having a comparator section mapped to an array section, the CAM structure transmitting a first output signal for generating the dependency data stored in the dependency matrix and a second output signal for generating the physical register address data stored in the IWB;

wherein the instructions have a first predetermined maximum number of register source fields (RS fields) having an architectural addresses for instruction input data and at least a maximum number of one register destination fields (RD fields) having architectural addresses for instruction output data; and

wherein the CAM structure further comprises:

the comparator section including a register dependency checker (RDC) for comparing an RS field of a fetched instruction to an RD field of an in-flight instruction; and

the first output signal including a hit detect signal indicative of the architectural address of the RD field which is generated when a match, i.e. hit, between the RS field and the RD field is detected by the RDC, the hit detect signal additionally being transmitted to the array section of the CAM structure.

44. (new) The computer system of claim 43 wherein the CAM structure further comprises:

the array section including an in-flight physical register mapper (IPM); and

the second output signal including an output signal indicative of the physical register address of the RD field which is generated when the IPM receives the hit detect signal.

45. (new) The computer system of claim 43 wherein the CAM structure further comprises a plurality of the first predetermined number of CAM structures, each CAM structure being dedicated to a single RS field.

46. (new) The computer system of claim 45 wherein the hit detect signals of each CAM structure are OR-ed together in an ORing unit to generate the dependency data.

47. (new) The computer system of claim 46 further comprising:

the plurality of physical registers including a queue area for storing a second predetermined maximum number of in-flight instructions of the instruction set; and

the fetch unit fetching a bundle having a third predetermined fixed number of instructions from the instruction set.

48. (new) The computer system of claim 47 further comprising an intra dependency checker having an output signal indicative of the dependencies between the instructions in the bundle, the output signal of the intra dependency checker being combined with the OR-ed output signals of the hit detect signals to generate the dependency data.

49. (new) The computer system of claim 48 wherein the RDC further comprises:

a plurality of the second predetermined number of entry structures, each entry structure including a plurality of the third predetermined number of comparators, each comparator dedicated to a single instruction in the bundle; wherein each RD field of the in-flight instructions is compared by a single entry structure to all instructions in the bundle substantially simultaneously.

50. (new) The computer system of claim 49 wherein the hit detect signal further comprises an array of the second predetermined number by the third predetermined number in size, which includes the output of hits detected for each of the entry structures compared to each of the instructions in the bundle.

51. (new) A method for performing out of order processing on an instruction set having a plurality of instructions with a superscalar processor, the method comprising:

fetching the instructions from the instruction;

renaming architectural registers associated with the instructions to physical registers during the processing of the instruction set;

transmitting a first output signal from a content addressable memory (CAM) structure for generating dependency data of in-flight instructions of the instruction set;

transmitting a second output signal from the CAM structure for generating physical register address data necessary to execute the in-flight instructions when the dependencies of the in-flight instructions have been removed;

storing the dependency data in a dependency matrix;

storing the physical register address data in an instruction wait buffer;

scheduling the in-flight instructions for execution based on the dependency data and the physical register address data; and

executing the in-flight instructions;

wherein the instructions have a first predetermined maximum number of register source fields (RS fields) having an architectural addresses for instruction input data and at least a maximum number of one register destination fields (RD fields) having architectural addresses for instruction output data; and wherein the transmitting the first output signal further comprises:

comparing an RS field of a fetched instruction to an RD field of an in-flight instruction within a comparator section of a CAM structure; and

generating a hit detect signal indicative of the architectural address of the RD field when the RS field and the RD field match, i.e. hit, and

transmitting the hit detect signal additionally to an array section of the CAM structure.

52. (new) The method of claim 51 wherein transmitting the second output signal further comprises generating the second output signal when the array section receives the hit detect signal, the second output signal being indicative of the physical register address of the RD field.

53. (new) The method of claim 52 wherein the CAM structure further comprises a plurality of the first predetermined number of CAM structures, each CAM structure being dedicated to a single RS field.

54. (new) The method of claim 53 further comprising logically ORing the hit detect signals of each CAM structure to generate the dependency data.

55. (new) The method of claim 54 further comprising:

fetching a bundle of instructions from the instruction set;

generating an intra-dependency signal indicative of intra-dependencies of instructions within the bundle;

combining the intra-dependency signal with the OR-ed output signals of the hit detect signals to generate the dependency data.